COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
1807-1856-1920

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

15 DAAG53-76-C-Ø138,
DARPA Order-32Ø6

12 4Ø

14 TR-780
DAAG-53-76C-0138

11 July 1979

Accession For	
NTIS GRA&I	✓
DDC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist A	Avail and/or special

6

A DISTANCE TRANSFORM FOR IMAGES REPRESENTED BY QUADTREES.

10 Hanan/Samet
Computer Science Department
University of Maryland
College Park, MD 20742

9 Technical rept.,

ABSTRACT

The concept of distance used in binary array representations of images is adapted to a quadtree representation. The Chessboard distance metric is shown to be particularly suitable for the quadtree. A Chessboard distance transform for a quadtree is defined as the minimum distance in the plane from each BLACK node to the border of a WHITE node. An algorithm is presented which computes this transform by only examining the BLACK node's adjacent and abutting neighbors and their progeny. Analysis of the algorithm shows that its average execution time is proportional to the number of leaf nodes in the quadtree.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper. The author has also benefited greatly from discussions with Charles R. Dyer, Azriel Rosenfeld, and M. Shneier. He also thanks Pat Young for drawing the figures.

409022 JOB

1. Introduction

In this paper we continue our investigation [2,9-13] into the usefulness of the quadtree [3-7] as a data structure for image processing. We investigate the concept of distance [1,8] and attempt to formulate a definition and metric which are applicable to quadtrees. Such a concept is useful for computing properties of an image such as its skeleton as well as for applying operations such as propagation and shrinking [8].

Section 2 contains a brief definition of the representation used. In Section 3 we present a brief review of the concept of distance and its application to the quadtree. In addition, we discuss some properties of a quadtree which will be seen to limit the amount of work necessary in computing the distance transform of a quadtree. Sections 4-6 contain an algorithm for computing the Chessboard distance transform for an image represented by a quadtree and an analysis of its execution time. The algorithm is presented using a variant of ALGOL 60.

2. Definitions and notation

We assume that the given image is a 2^n by 2^n array of unit square "pixels". The quadtree is an approach to image representation based on a successive subdivision of the array into quadrants. In essence, we repeatedly subdivide the array into quadrants, subquadrants,... until we obtain blocks (possibly single pixels) which consist entirely of 1's or Ø's. This process is represented by a tree of out-degree 4 in which the root node represents the entire array, the four sons of the root node represent the quadrants, and the terminal nodes correspond to those blocks of the array for which no further subdivision is necessary. For example, Figure 1b is a block decomposition of the region in Figure 1a while Figure 1c is the corresponding quadtree. In general, BLACK and WHITE square nodes represent blocks consisting entirely of 1's and Ø's respectively. Circular nodes, also termed GRAY nodes, denote non-terminal nodes.

Each node in a quadtree is stored as a record containing seven fields. The first five fields contain pointers to the node's father and its four sons, labeled NW, NE, SE, and SW. Given a node P and a son I, these fields are referenced as FATHER(P) and SON(P,I) respectively. At times it is useful to use the function SONTYPE(P) where SONTYPE(P)=Q iff SON(FATHER(P),Q)=P. The sixth field, NODETYPE, describes the contents of the block of the image which the node represents--i.e., BLACK, WHITE, or GRAY. The seventh field, DIST, indicates the distance to the nearest WHITE node according to the specified distance metric. This field is only meaningful for BLACK nodes. A WHITE node is said to have distance zero.

Let the four sides of a node's block be called its N, E, S, and W sides. They are also termed its boundaries and at times we speak of them as if they are directions. Figure 2 shows the relationship between the quadrants of a node's block and its boundaries. The specification of the spatial relationships between the various sides is facilitated by use of the functions OPSIDE, CSIDE, and CCSIDE. OPSIDE(B) is a side facing side B; e.g., OPSIDE(N)=S. *CSIDE(B)* and CCSIDE(B) correspond to the sides adjacent to side B in the clockwise and counterclockwise directions respectively; e.g., CSIDE(N)=E and CCSIDE(N)=W. We also define the following predicates and functions to aid in the expression of operations involving a block's quadrants and boundaries. ADJ(B,I) is true if and only if quadrant I is adjacent to boundary B of the node's block, e.g., ADJ(E,SE) is true. REFLECT(B,I) yields the SONTYPE value of the block of equal size that is adjacent to side B of a block having SONTYPE value I; e.g., REFLECT(N,SE)=NE, REFLECT(E,SE) =SW, REFLECT(S,SE)=NE, and REFLECT(W,SE)=SW. COMMONSIDE(Q1,Q2) indicates the boundary of the block containing quadrants Q1 and Q2 that is common to them (if Q1 and Q2 are not adjacent brother quadrants, then the value of COMMONSIDE(Q1,Q2) is undefined); e.g., COMMONSIDE(SE,SW)=S while COMMONSIDE(NW,SE) is undefined. QUAD(S1,S2) is the quadrant bounded by boundaries S1 and S2 (if S1 and S2 are not adjacent boundaries, then QUAD(S1,S2) is undefined); e.g., QUAD(S,E)=SE while QUAD(S,N) is undefined. Similarly, OPQUAD(QUAD(S1,S2)=QUAD(OPSIDE(S1), OPSIDE(S2)).

For a quadtree corresponding to a 2^n by 2^n array we say that the root is at level n, and that a node at level i is at a distance of n-i from the root of the tree. In other words, for a node at level i, we must ascend n-i FATHER links to reach the root of the tree. Note that the farthest node from the root of the tree is at level $\geq \emptyset$. A node at level $\emptyset$ corresponds to a single pixel in the image. Also we say that a node is of size 2^S if it is found at level S in the tree--i.e., it has a side of length 2^S.

3. Distance

For an image represented by a binary array, we can define a function d that takes pairs of points into non-negative numbers. It is called a metric, or a distance function, if for all points p, q, and r the following relations are satisfied.

(1) $d(p,q) \geq 0$, and $d(p,q)=\emptyset$ if and only if $p=q$ (positive definiteness)

(2) $d(q,p)=d(p,q)$ (symmetry)

(3) $d(p,r) \leq d(p,q)+d(q,r)$ (triangle inequality)

Given the points $p=(p_x,p_y)$ and $q=(q_x,q_y)$ we now examine some of the more common metrics. The most commonly used metric is the Euclidean distance

$$d_E(p,q) = \sqrt{(p_x-q_x)^2+(p_y-q_y)^2}$$

Two other metrics which are used in image processing are the Absolute Value metric, or the City Block distance,

$$d_A(p,q) = |p_x-q_x| + |p_y-q_y|$$

and the Maximum Value metric, or the Chessboard distance,

$$d_M(p,q) = \max\{|p_x-q_x|,|p_y-q_y|\}$$

The set of points, q, having $d_E(p,q) \leq t$ are those points contained in a circle centered at p having radius t. Similarly, $d_A(p,q) \leq t$ yields a diamond, centered at p, with side length $t\sqrt{2}$, and $d_M(p,q) \leq t$ yields a square, centered at p, with side length 2t.

For an image represented by a quadtree we use the same metrics. The only difference is that the points for which the metrics are defined are centers of blocks. We also define the distance transform T for a quadtree to be a function that yields for each BLACK block in the quadtree the distance (in the chosen metric) from the center of the block to the nearest point which is on a BLACK-WHITE border. More formally, letting x be the center of a BLACK block B, z be a point on the border of a WHITE block W, and only using F intermediately as the distance to a particular WHITE block's border, we have:

$$F(B,W) = \min_{z} d(x,z)$$

$$T(B) = \min_{W} F(B,W)$$

We say that T of a WHITE block is zero and that the border of the space represented by the quadtree of the image is BLACK. Notice that the distance transform is not defined in terms of a center to center distance. This is done to avoid a bias against large size WHITE adjacent blocks and moreover it will be seen to enable us to restrict the number of nodes visited while computing it.

Given blocks P and Q, we say that Q is a _neighbor_ of P when both of the following conditions are satisfied.

(1) P and Q share a common border, even if only a corner.

(2) If Q is a BLACK or WHITE block, then its size is greater than or equal to that of P, while if it is a GRAY block, then it and P are of equal size.

For example, block R in Figures 1b and 1c has neighbors O, NN, OO, Q, HH, PP, MM, and S. A block has a maximum of eight neighbors, in which case they are all of equal size (e.g., block O in Figure 1b), and a minimum of five neighbors. The minimum is obtained by observing that a node cannot be adjacent to two nodes of greater size or opposite sides (e.g., given node P, and nodes Q and R adjacent to its east and west sides respectively, then nodes Q and R cannot be both greater in size than P). Thus a node can have at most two larger sized neighbors adjacent to its non-opposite sides and two of these neighbors can subsume at most three additional neighbors, thereby requiring at least three more neighbors (e.g., for node D in Figure 1b, AA subsumes the NW, N, and NE neighbors; C subsumes the W and SW neighbors; the remaining neighbors are E, F, and DD in directions S, SE, and E respectively. We can now prove the following theorem which aids in understanding the amount of work involved in computing the distance transform for any metric.

Theorem 1: For any BLACK block in the image, its neighbors cannot all be BLACK.

Proof: One of the neighbors of the block, say P, must be GRAY or WHITE since otherwise merging would have taken place and P would not be in the image (i.e., P would be part of a bigger BLACK block).

Q.E.D.

Theorem 1 makes the Chessboard distance metric especially attractive. It means that for a BLACK block of size 2^S, say P, the center of the WHITE block whose border is nearest to P (hereinafter referred to as the nearest WHITE block) must be found at a distance of $<3\cdot 2^{S-1}$--i.e., within a square centered at P of side length $3\cdot 2^{S}$. In fact, the worst case arises when the nearest WHITE block is a block of minimum size (i.e., a single pixel) adjacent to the furthest boundary of P's neighboring GRAY block (e.g., WHITE block EE with respect to BLACK block N in Figure 1b). Notice that none of P's neighboring BLACK blocks need be taken into consideration in computing P's Chessboard distance transform since the value would have to be at least $3\cdot 2^{S-1}$ (e.g., BLACK blocks O, Q, and R with respect to BLACK block N in Figure 1b). Thus Theorem 1 means that when computing the Chessboard distance transform of a quadtree, for each node corresponding to a BLACK block we only need to consider its GRAY neighboring nodes. Figure 3a illustrates the worst case in terms of the number of blocks that need to be examined--i.e., BLACK block 1 is surrounded by rings of BLACK blocks of decreasing size.

Theorem 1 can also be used to constrain the amount of work needed to compute other distance transforms. In the case of the Euclidean distance transform given BLACK node P of size 2^{S}, the nearest WHITE block is at a distance $<3\cdot 2^{S-1}\cdot\sqrt{2}$. Similarly, when a City Block distance transform is used the maximum distance is $<3\cdot 2^{S}$. These values are all derived analogously.

Unfortunately, we cannot say that larger sized neighbors need not be taken into consideration when computing the Euclidean and City Block distance transforms. That this is true can be seen by examining Figure 4 which illustrates the regions within which Theorem 1 stipulates that the nearest WHITE block be found. For example, given BLACK block A of size 2^S, in the case of both the Euclidean and City Block distance, block B may be the nearest WHITE block to block A. This may require visiting an eastern neighbor of block A of size 2^{S+1}. On the other hand, when Chessboard distance is used, block C is the nearest to block A and, in fact, no neighboring blocks of greater size ever need to be visited. Thus we see that the Euclidean and City Block distances may lead to more than eight neighboring blocks of equal size being visited or even to blocks of greater size.

Note that our definition of distance treats non-existent neighbors (i.e., on the other side of the border of the space represented by the quadtree) as BLACK and of equal size. This is consistent with the definition of the Chessboard distance transform as yielding for each BLACK node in the quadtree the distance from the center of the block to the nearest point which is on the border between a BLACK and a WHITE node.

In the remainder of this paper we focus our attention on the Chessboard distance transform due to its computational simplicity. This simplicity arises from the property of the Chessboard distance metric that the set of points q such that $d(p,q) \leq t$ is a square, rather than a circle or a diamond as is true for the Euclidean and City Block distance metrics respectively.

4. Algorithm

The Chessboard distance transform algorithm traverses the quadtree in postorder (i.e., the sons of a node are visited first). Recalling the definition of a neighbor given in Section 3, we have that for each BLACK node of size 2^S, its eight neighbors in the N, NE, E, SE, S, SW, W, and NW directions may have to be explored in determining the nearest WHITE node. If any of the neighbors is WHITE, then the minimum distance is 2^{S-1} and we cease processing. Neighboring BLACK nodes do not affect the value of the Chessboard distance transform since they result in a minimum transform value of $3 \cdot 2^{S-1}$ which exceeds the theoretical maximum. Thus the heart of the algorithm lies in processing GRAY nodes.

The main procedure is termed CHESSBOARD_DIST and is invoked with a pointer to the root of the quadtree representing the image and an integer corresponding to the log of the diameter of the image (e.g., n for a 2^n by 2^n image array). CHESSBOARD_DIST traverses the tree and controls the exploration of the eight neighbors of each BLACK node.

GTEQUAL_ADJ_NEIGHBOR locates a neighbor along a specified direction (e.g., N, E, S, or W). If the node is on the edge of the image, then no neighbor exists in the specified direction and NULL is returned (e.g., the western neighbor of node C in Figure 1b). If the node is not on the edge of the image and no neighboring BLACK or WHITE node exists, then a pointer to a GRAY node of equal size is returned (.e.g., the eastern neighbor of node 1 in Figure 3a). In such a case, procedure DIST_ADJACENT continues the search by examining the subquadrants of the adjacent GRAY node. We first examine the nodes corresponding to the subquadrants adjacent to the side of the node being processed (e.g., subquadrants NW and SW of the eastern neighbor of node 1 in Figure 3a). If either node is WHITE, then a closest WHITE node in the specified direction has been found. If both nodes are GRAY, then we recursively apply DIST_ADJACENT to the corresponding subquadrants. If both nodes are BLACK, then we examine the remaining two subquadrants in a similar manner (e.g., subquadrants NE and SE of the eastern neighbor of node 1 in Figure 3a).

GTEQUAL_CORNER_NEIGHBOR locates a neighbor adjacent to a specified corner (e.g., NE, SE, SW, or NW). If the node is on the edge of the image, then no neighbor exists in the specified corner and NULL is returned (e.g., the NW neighbor of node C in Figure 1b). If the node is not on the edge of the image and no neighboring BLACK or WHITE node exists, then a pointer to a GRAY node of equal size is returned (e.g., the NE neighbor of node 1 in Figure 3a). In such a case, procedure DIST_CORNER continues the search by examining the subquadrants of the adjacent GRAY node. We first examine the nodes corresponding to the subquadrant which is adjacent to the corner of the node being processed (e.g., subquadrant SW of the NE neighbor of node 1 in Figure 3a). If the node is WHITE, then a closest node in the specified direction has been found. If the node is GRAY, then we recursively apply DIST_CORNER to the subquadrant. If the node is BLACK, then we recursively examine the three remaining subquadrants in the following manner. We apply DIST_ADJACENT to the BLACK node's adjacent subquadrants in a direction which is adjacent to the BLACK node (e.g., DIST_ADJACENT is applied to the NW and SE subquadrants of the NE neighbor of node 1 in Figure 3a). We also apply DIST_CORNER to the non-adjacent subquadrant (e.g., the NE subquadrant of the NE neighbor of node 1 in Figure 3a).

As an example of the application of the algorithm, consider the region given in Figure 1a. Figure 1b is the corresponding block decomposition while Figure 1c is its quadtree representation. All of the BLACK nodes have labels ranging from A to R while the WHITE nodes have labels ranging between AA and PP. The GRAY nodes have labels ranging between 1 and 11.

The BLACK nodes are labeled in the order in which their adjacencies are explored by CHESSBOARD_DIST. Figure 1d contains the Chessboard distance transform corresponding to Figure 1b.

```
procedure CHESSBOARD_DIST(P,LEVEL);
/* Given a quadtree rooted at node P spanning a 2↑LEVEL by
   2↑LEVEL space, find the Chessboard distance of each BLACK
   node to its closest WHITE node.  WHITE nodes are assigned
   distance Ø */
   begin
        value node P;
        node Q;
        value integer LEVEL;
        integer C;
        quadrant I;
        direction D;
        if GRAY(P) then
            begin
                for I in {'NW','NE','SW','SE'} do
                    CHESSBOARD_DIST(SON(P,I),LEVEL-1);
            end
        else if BLACK(P) then
            begin
                D←'N';
                C←2↑LEVEL;
                do
                    begin
                        Q←GTEQUAL_ADJ_NEIGHBOR(P,D);
                        D←if NULL(Q) or BLACK(Q) then C
                          else if WHITE(Q) then Ø
                          else DIST_ADJACENT(Q,QUAD(OPSIDE(D),CCSIDE(D)),
                                             QUAD(OPSIDE(D),CSIDE(D)),
                                             2↑(LEVEL-1),Ø,C);
                if C≠0 then
                  begin
                      Q←GTEQUAL_CORNER_NEIGHBOR(P,QUAD(D,CSIDE(D)));
                      D←if NULL(Q) or BLACK(Q) then C
                        else if WHITE(Q) then Ø
                        else DIST_CORNER(Q,D,CSIDE(D),2↑(LEVEL-1),
                                          Ø,C);
                      D←CSIDE(D);
                  end;
            end
            until C=Ø or D='N';
        DIST(P)←C+2↑(LEVEL-1);
    end
  else DIST(P)←Ø;  /* a WHITE node */
end;
```

```
node procedure GTEQUAL_ADJ_NEIGHBOR(P,D);
/*  Return the neighbor of node P in horizontal or vertical direc-
    tion D.  If such a node does not exist, then return NULL  */
begin
    value node P;
    node Q;
    value direction D;
    if not NULL(FATHER(P)) and ADJ(D,SONTYPE(P)) then
           /* Find a common ancestor */
        Q←GTEQUAL_ADJ_NEIGHBOR(FATHER(P),D)
    else Q←FATHER(P);
    /* Follow the reflected path to locate the neighbor */
    return (if not NULL(Q) and GRAY(Q) then SON(Q,REFLECT(D,SONTYPE(P)))
              else Q);
end;

integer procedure DIST_ADJACENT(P,Q1,Q2,W,B,C);
/*  Given a subquadtree rooted at node P spanning a 2·W by 2·W space,
    the distance of the closest WHITE node to the border formed
    by quadrants Q1 and Q2 of P.  B is a lower bound for the
    distance.  C is the minimum distance obtained so far  */
begin
    value node P;
    value quadrant Q1,Q2;
    value integer B,C,W;
    return (if B≥C then C  /* The minimum has already been found  */
        else if WHITE(P) then B
        else if BLACK(P) then C
        else if BLACK(SON(P,Q1)) and BLACK(SON(P,Q2)) then
            DIST_ADJACENT(SON(P,OPQUAD(Q1)),Q1,Q2,W/2,B+W
                          DIST_ADJACENT(SON(P,OPQUAD(Q2)),Q1,Q2,
                                        W/2,B+W,C))
        else DIST_ADJACENT(SON(P,Q2),Q1,Q2,W/2,B,
                           DIST_ADJACENT(SON(P,Q1),Q1,Q2,W/2,B,C)));
end;

node procedure GTEQUAL_CORNER_NEIGHBOR(P,C);
/*  Return the neighbor of node P in diagonal direction C.  If
    such a node does not exist, then return NULL  */
begin
    value node P;
    node Q;
    value quadrant C;
    if not NULL(FATHER(P)) and SONTYPE(P)≠OPQUAD(C) then
        if SONTYPE(P)=C then Q←GTEQUAL_CORNER_NEIGHBOR(FATHER(P),C)
        else Q←GTEQUAL_ADJ_NEIGHBOR(FATHER(P),COMMONSIDE(SONTYPE(P),C))
    else Q←FATHER(P);
    /*  Follow the opposite path to locate the neighbor  */
    return (if not NULL(Q) and GRAY(Q) then SON(Q,OPQUAD(SONTYPE(P)))
            else Q);
end;
```

```
integer procedure DIST_CORNER(P,D1,D2,W,B,C);
/*  Given a subquadtree rooted at node P spanning a 2·W by 2·W space,
    return the distance of the closest WHITE node to the corner
    formed by quadrant OPQUAD(QUAD(D1,D2)) of P.  B is a lower bound
    for the distance.  C is the minimum distance obtained so far. */
begin
    value node P;
    value direction D1,D2;
    value integer B,C,W;
    integer TEMP;
    if C≥D then return (C)  /* The minimum has already been found */
    else if WHITE(P) then return(B)
    else if BLACK(P) then return(C)
    else
        begin
            TEMP←DIST_CORNER(SON(P,OPQUAD(QUAD(D1,D2))),D1,D2,W/2,B,C);
            TEMP←DIST_ADJACENT(SON(P,QUAD(D1,OPSIDE(D2))),
                               OPQUAD(QUAD(D1,D2)),
                               QUAD(OPSIDE(D1),D2),W/2,B+W,TEMP);
            TEMP←DIST_ADJACENT(SON(P,QUAD(OPSIDE(D1),D2)),
                               QUAD(D1,OPSIDE(D2)),
                               OPQUAD(QUAD(D1,D2)),W/2,B+W,TEMP);
            TEMP←DIST_CORNER(SON(P,QUAD(D1,D2),D1,D2,W/2,B+W,TEMP);
            return(TEMP);
    end;
end;
```

5. Analysis

The running time of the Chessboard distance transform computation algorithm, measured by the number of nodes visited, depends on the time spent locating WHITE nodes and on the size of the quadtree. Theorem 1 guarantees that we only need to examine a maximum of eight neighbors and find their WHITE progeny with a minimum distance. This is accomplished by procedures GTEQUAL_ADJ_NEIGHBOR, DIST_ADJACENT, GTEQUAL_CORNER_NEIGHBOR, and DIST_CORNER. For each BLACK node, each procedure is invoked a maximum of four times. The amount of work performed by these procedures is obtained by considering the number of nodes that are visited whenever a neighbor and its progeny are searched. Recall that we must find the neighbor, and if it is GRAY, then visit the nearest WHITE node of smaller size. In the worst case we are at level n-1, with a GRAY neighbor, and all the nearest WHITE nodes are at level Ø and at the maximum distance. In such a case we will have to visit $2^{n+1}-2$ nodes in the case of a horizontal or vertical neighbor, and $2^{n+2}-2(2n+1)$ nodes in the case of a diagonal neighbor. For example, consider Figure 5, where n=3 and nodes are labeled in the order in which the neighbors of node A and their progeny have been visited starting with the northern neighbor and proceeding clockwise; we visit the north neighbor and its progeny of the block labeled A (i.e., blocks B, C, D, E, F, G, H, I, J, K, and four GRAY nodes on of which is a common ancestor) and the NE neighbor and its progeny (i.e., blocks L, M, N, O, P, Q, R, S, T, U, V, W, X, and five GRAY nodes one of which is a common ancestor).

In the following we analyze the average execution time of the Chessboard distance computation algorithm. Our analysis assumes a 2^n by 2^n random image in the sense that a node is equally likely to appear in any position and level in the quadtree. This means that we assume that all configurations of adjacent nodes of varying sizes have equal probability. This is different from the more conventional notion of a random image which implies that every block at level Ø (i.e., pixel) has an equal probability of being BLACK or WHITE. Such an assumption would lead to a very low probability of any nodes corresponding to blocks of size larger than 1. Clearly, for such an image the quadtree is the wrong representation. We first derive the average number of nodes visited by GTEQUAL_ADJ_NEIGHBOR and DIST_ADJACENT. Next we perform a similar analysis for the number of nodes visited by GTEQUAL_CORNER_NEIGHBOR and DIST_CORNER.

Lemma 1: The average of the minimum number of nodes visited by each invocation of GTEQUAL_ADJ_NEIGHBOR and DIST_ADJACENT is 6.

Proof: Given a node P at level i and a horizontal or vertical direction D, there are $2^{n-i}(2^{n-i}-1)$ possible positions for node P and a neighbor at level i and direction D. Of these $2^{n-i}(2^{n-i}-1)$ neighbor pairs, $2^{n-i} \cdot 2^{\emptyset}$ have their nearest common ancestor at level n, $2^{n-i} \cdot 2^{1}$ at level n-1,..., and $2^{n-i} \cdot 2^{n-i-1}$

at level i+1. For each node at level i having a common ancestor at level j, the maximum number of nodes that will be visited by GTEQUAL_ADJ_NEIGHBOR and DIST_ADJACENT is

$$(j-i) + (j-i) + 4\sum_{k=0}^{i-1} 2^k = 2(j-i-2) + 2^{i+2}$$

This is obtained by observing that the common ancestor is at a distance of j-i and that a node at level i has a maximum of 2^i WHITE nodes at a maximum distance from it in a specified direction (all appearing at level Ø). For example, consider the N neighbor of node 1 in Figure 3a where i=2. The sum reflects the fact that for each GRAY node at level k>Ø, four nodes must be visited (two BLACK and two GRAY). Assuming that node P is equally likely to occur at any level i and at any of the $2^{n-i}(2^{n-i}-1)$ positions at level i, then the average of the maximum number of nodes visited by GTEQUAL_ADJ_NEIGHBOR and DIST_ADJACENT is

$$\frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} 2^{n-i}\cdot 2^{n-j}(2(j-i-2)+2^{i+2})}{\sum_{i=0}^{n-1} 2^{n-i}(2^{n-i}-1)} \qquad (1)$$

(1) can be rewritten to yield

$$\frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1-i} 2^{2n-2i-1-j}(2(j-1)+2^{i+2})}{\sum_{i=1}^{n} 2^{i}(2^{i}-1)} \qquad (2)$$

The numerator of (2) can be simplified as follows:

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-1-i} 2^{2n-2i-1-j}(2(j-1)+2^{i+2}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1-i} ((j-1)2^{2n-2i-j}+2^{2n-i+1-j})$$

$$= \sum_{i=0}^{n-1} 2^{2n-2i} \sum_{j=0}^{n-1-i} \left(\frac{j}{2^j} - \frac{1}{2^j}\right) + \sum_{i=0}^{n-1} 2^{2n-i+1} \sum_{j=0}^{n-i-1} \frac{1}{2^j} \qquad (3)$$

But $$\sum_{j=0}^{n-1-i} \frac{j}{2^j} = 2 - \frac{n+1-i}{2^{n-1-i}} \qquad (4)$$

Also $$\sum_{j=0}^{n-1-i} \frac{1}{2^j} = 2\left(1 - \frac{1}{2^{n-i}}\right) \qquad (5)$$

Substituting (4) and (5) into (3) yields

$$\sum_{i=0}^{n-1} \left(2^{2n-2i}\left(2 - \frac{n+1-i}{2^{n-1-i}} - 2\left(1- \frac{1}{2^{n-i}}\right)\right) + 2^{2n-i+1}\cdot 2\left(1 - \frac{1}{2^{n-i}}\right)\right)$$

$$= \sum_{i=0}^{n-1} (2^{n-i+1}(2^{n-i}-(n+1-i))-2^{n-i+1}(2^{n-i}-1)+2^{n+2}(2^{n-i}-1))$$

$$= \sum_{i=0}^{n-1} (2^{2n-2i+1}-n\cdot 2^{n-i+1}-2^{n-i+1}+i\cdot 2^{n-i+1}-2^{2n-2i+1}$$

$$+2^{n-i+1}+2^{2n-i+2}-2^{n+2})$$

$$= \sum_{i=0}^{n-1} (2^{n-i+2}-n\cdot 2^{n-i+1}+i\cdot 2^{n-i+1}-2^{n+2})$$

$$= 2^{2n+2} \sum_{i=0}^{n-1} \frac{1}{2^i} - n\cdot 2^{n+1} \sum_{i=0}^{n-1} \frac{1}{2^i} + 2^{n+1} \sum_{i=0}^{n-1} \frac{i}{2^i} - n\cdot 2^{n+2}$$

$$= 2^{2n+2}\cdot 2\left(1 - \frac{1}{2^n}\right) - n\cdot 2^{n+1}\cdot 2\left(1 - \frac{1}{2^n}\right) + 2^{n+1}\left(2 - \frac{n+1}{2^{n-1}}\right) - n\cdot 2^{n+2}$$

$$= 2^{n+3}(2^n-1)-4n(2^n-1)+4(2^n-(n+1))-n\cdot 2^{n+2}$$

$$= 2^{2n+3}-(2n+1)\cdot 2^{n+2}-4 \qquad (6)$$

The denominator of (2) can be simplified as follows:

$$\sum_{i=1}^{n} 2^i(2^i-1) = \sum_{i=0}^{n} (2^{2i}-2^i)$$

$$= \sum_{i=0}^{n} 4^i - \sum_{i=0}^{n} 2^i$$

$$= \frac{4^{n+1}-1}{3} - (2^{n+1}-1)$$

$$\text{or} \quad \sum_{i=1}^{n} 2^i(2^i-1) = \tfrac{1}{3}(2^{2n+2}-3\cdot 2^{n+1}+2) \qquad (7)$$

Substituting (6) and (7) into (2) yields

$$\frac{2^{2n+3}-(2n+1)\cdot 2^{n+2}-4}{\frac{1}{3}(2^{2n+2}-3\cdot 2^{n+1}+2)} = 6 - \frac{12\ (n-1)\cdot 2^n+1}{2^{2n+1}-3\cdot 2^n\ +1}$$

$$< 6$$

Q.E.D.

<u>Lemma 2</u>: The average of the maximum number of nodes visited by each invocation of GTEQUAL_CORNER_NEIGHBOR and DIST_CORNER is less than 8.

<u>Proof</u>: Given a node P at level i and a diagonal direction C, there are $(2^{n-i}-1)^2$ possible positions for node P and a neighbor at level i in direction C. Of these $(2^{n-i}-1)^2$ neighbor pairs, $4^0\cdot(2\cdot(2^{n-i}-1)-1)$ have their nearest common ancestor at level n, $4^1(2\cdot(2^{n-i-1}-1)-1)$ at level n-1,... and $4^{n-i-1}\cdot(2\cdot(2^{n-i-(n-i-1)}-1)-1)$ at level i+1. In order to see this, consider Figure 6 where a grid is shown for n=3. If all BLACK and WHITE nodes are at level Ø, then for a neighbor in the NE direction we see that nodes along the fifth row and

fourth column have their nearest common ancestor at level 3 (i.e., 13 nodes labeled 1-13). Continuing the process for the NW, NE, SW, and SE quadrants of Figure 6 we find that all neighbor pairs contained exclusively within these quadrants have their nearest common ancestor at a level ≤ 2. In particular, for the NW quadrant, nodes along the third row and second column have their nearest common ancestor at level 2 (i.e., 5 nodes labeled 14-18). The NE, SW, and SE quadrants are analyzed in a similar manner. This process is applied to the four subquadrants of the quadrants to obtain the neighbor pairs whose nearest common ancestor is at level 1. Note that we had to consider every row in the image when analyzing diagonal neighbor pairs whereas we only needed to consider one row or column when analyzing neighbor pairs in the N, E, S, and W directions. This is necessary because for diagonal neighbors each row in the image has a different number of neighbor pairs with a common ancestor at a given level while this number is constant for each row or column when considering neighbor pairs in the horizontal and vertical directions.

For each node at level i having a nearest common ancestor at level j, the maximum number of nodes that will be visited by GTEQUAL_CORNER_NEIGHBOR and DIST_CORNER is

$$(j-i)+(j-i) + \sum_{k=0}^{i-1} (4+2-4 \sum_{\ell=0}^{k-1} 2^{\ell}) = 2(j-3i-4)+2^{i+3}$$

This is obtained by observing that the common ancestor is at a distance of j-i and that a node at level i has a maximum of $2^{i+1}-1$ WHITE nodes at a maximum distance from it in a specified diagonal direction (all appearing at level Ø). For example, consider the NE neighbor of node 1 in Figure 3a where i=2. The sum reflects the fact that for each GRAY node at level $k>0$, four nodes must be visited (one BLACK and three GRAY). Two of the blocks spanned by the GRAY nodes are visited by use of DIST_ADJACENT while the block corresponding to the remaining GRAY node is visited by recursively reapplying DIST_CORNER. Assuming that node P is equally likely to occur at any level i and at any of the $(2^{n-i}-1)^2$ positions at level i, then the average of the maximum number of nodes visited by GTEQUAL_CORNER_NEIGHBOR and DIST_CORNER is

$$\frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} 4^{n-j}\cdot(2\cdot(2^{n-i-(n-j)}-1)-1)(2(j-3i-4)+2^{i+3})}{\sum_{i=0}^{n-1}(2^{n-i}-1)^2}$$

$$= \frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} (2^{2n-j-i+1}-3\cdot 2^{2n-2j})(2(j-3i-4)+2^{i+3})}{\sum_{i=0}^{n-1}(2^{n-i}-1)^2} \qquad (8)$$

(8) can be rewritten to yield

$$\frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1-i} (2^{2n-2i-j}-3\cdot 2^{2n-2i-2-2j})(2(j-2i-3)+2^{i+3})}{\sum_{i=1}^{n}(2^{i}-1)^2} \qquad (9)$$

The numerator of (9) can be simplified as follows:

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1-i} (2^{2n-2i-j}-3\cdot 2^{2n-2i-2-2j})(2(j-2i-3)+2^{i+3})$$

$$= \sum_{i=0}^{n-1} 2^{2n+1}\sum_{j=0}^{n-1-i} (2^{-2i-j}-3\cdot 2^{-2i-2j-2})(j-2i-3+2^{i+2})$$

$$= \sum_{i=0}^{n-1} 2^{2n+1-2i}\sum_{j=0}^{n-1-i}\left(\frac{j-2i-3+2^{i+2}}{2^{j}} - 3\cdot\frac{j-2i-3+2^{i+2}}{2^{2j+2}}\right) \qquad (10)$$

But $\sum_{j=0}^{n-1-i} \frac{j}{2^j} = 2 - \frac{n+1-i}{2^{n-1-i}}$ (11)

$$\sum_{j=0}^{n-1-i} \frac{-2i-3+2^{i+2}}{2^j} = (-4i-6+2^{i+3})(1 - \frac{1}{2^{n-i}}) \qquad (12)$$

$$\sum_{j=0}^{n-1-i} \frac{j}{2^{2j+2}} = \frac{1}{4}(\frac{4}{9} - \frac{3(n-1-i)+4}{9\cdot 2^{2n-2-2i}}) \qquad (13)$$

$$\sum_{j=0}^{n-1-i} \frac{-2i-3+2^{i+2}}{2^{2j+2}} = \frac{1}{3}(-2i-3+2^{i+2})(1- \frac{1}{2^{2n-2i}}) \qquad (14)$$

Substituting (11), (12), (13), and (14) into (10) yields

$$\sum_{i=0}^{n-1} 2^{2n+1-2i}(2 - \frac{n+1-i}{2^{n-1-i}} + (-4i-6+2^{i+3})(1 - \frac{1}{2^{n-1}})$$
$$- \frac{3}{4}(\frac{4}{9} - \frac{3(n-1-i)+4}{9\cdot 2^{2n-2-2i}}) - (-2i-3+2^{i+2})(1 - \frac{1}{2^{2n-2i}}))$$

$$= \sum_{i=0}^{n-1} \frac{2^{2n+1-2i}}{3\cdot 2^{2n-2i}} (6\cdot 2^{2n-2i}-3\cdot 2^{n+1-i}(n+1-i)+3\cdot 2^{n-i}(2^{i+3}-4i-6)(2^{n-i}-1)$$
$$-2^{2n-2i}+3(n-1-i)+4+3\cdot(2i+3-2^{i+2})(2^{2n-2i}-1))$$

$$= \sum_{i=0}^{n-1} \frac{2}{3}(6\cdot 2^{2n-2i}-3n\cdot 2^{n+1-i}-3\cdot 2^{n+1-i}+3i\cdot 2^{n+i-1}+3\cdot 2^{2n-i+3}-12i\cdot 2^{2n-2i}$$
$$-18\cdot 2^{2n-2i}-3\cdot 2^{n+3}+12i\cdot 2^{n-i}+18\cdot 2^{n-i}-2^{2n-2i}+3n-3-3i+4$$
$$+6i\cdot 2^{2n-2i}+9\cdot 2^{2n-2i}-3\cdot 2^{2n-i+2}-6i-9+3\cdot 2^{i+2})$$

$$= \sum_{i=0}^{n-1} \frac{2}{3}(3\cdot 2^{2n-i+2}-(4+6i)\cdot 2^{2n-2i}-3\cdot 2^{n+3}+6\cdot(3i-n+2)\cdot 2^{n-i}$$
$$+3\cdot 2^{i+2}+3n-9i-8)$$

$$= \frac{2}{3}((3\cdot 2^{2n+2}+6(2-n)2^n)\sum_{i=0}^{n-1} \frac{1}{2^i} - 4\cdot 2^{2n} \sum_{i=0}^{n-1} \frac{1}{2^{2i}} - 6\cdot 2^{2n} \sum_{i=0}^{n-1} \frac{i}{2^{2i}}$$
$$+18\cdot 2^n \sum_{i=0}^{n-1} \frac{i}{2^i} + 12\sum_{i=0}^{n-1} 2^i-9\sum_{i=0}^{n-1} i-3n\cdot 2^{n+3}+3n^2-8n)$$

$$= \frac{2}{3}((3\cdot 2^{2n+2}+6(2-n)2^{n})\cdot 2\cdot (1 - \frac{1}{2^{n}})-4\cdot 2^{2n}\cdot \frac{4}{3}\cdot (1 - \frac{1}{2^{2n}})$$

$$-6\cdot 2^{2n}(\frac{4}{9} - \frac{(3(n-1)+4)}{9\cdot 2^{2n-2}}) + 18\cdot 2^{n}(2 - \frac{n+1}{2^{n-1}})+12\cdot (2^{n}-1)$$

$$- \frac{9}{2}n\cdot (n-1)-3n\cdot 2^{n+3}+3n^{2}-8n)$$

$$= \frac{2}{3}((3\cdot 2^{2n+2}+6(2-n)2^{n})\cdot 2(\frac{2^{n}-1}{2^{n}})-4\cdot 2^{2n}\ \frac{4}{3}(\frac{2^{2n}-1}{2^{2n}}) - \frac{8}{3}(2^{2n}-3n-1)$$

$$+18\cdot 2^{n}(\frac{2^{n}-n-1}{2^{n-1}})+12(2^{n}-1)- \frac{9}{2}n^{2} + \frac{9}{2}n -3n\cdot 2^{n+3}+3n^{2}-8n)$$

$$= \frac{2}{3}((6\cdot 2^{n+2}+12(2-n))(2^{n}-1) - \frac{16}{3}(2^{2n}-1) - \frac{8}{3} \cdot 2^{2n}+8n + \frac{8}{3} +36\cdot 2^{n}$$

$$-36n-36+12\cdot 2^{n}-12 - \frac{9}{2}n^{2} + \frac{9}{2}n -3n\cdot 2^{n+3}+3n^{2}-8n)$$

$$= \frac{2}{9}(48\cdot 2^{2n} + 144\cdot 2^{n} - 108n\cdot 2^{n} - \frac{9}{2}n^{2} - \frac{117}{2}n - 192) \qquad (15)$$

The denominator of (9) can be simplified as follows:

$$\sum_{i=1}^{n} (2^{i}-1)^{2} = \sum_{i=0}^{n} 2^{2i}-2\sum_{i=0}^{n} 2^{i} + \sum_{i=0}^{n} 1$$

$$= \sum_{i=0}^{n} 4^{i}-2(2^{n+1}-1)+n+1$$

$$= \frac{4^{n+1}-1}{3} - 2^{n+2}+2+n+1$$

$$\text{or} \quad \sum_{i=1}^{n} (2^{i}-1)^{2} = \frac{1}{3}(2^{2n+2}-3\cdot 2^{n+2}+3n+8) \qquad (16)$$

Substituting (15) and (16) into (9) yields

$$\frac{\frac{2}{9}(48\cdot 2^{2n}+144\cdot 2^{n}-108n\cdot 2^{n} - \frac{9}{2}n^{2} - \frac{117}{2}n-192)}{\frac{1}{3}(2^{2n+2}-3\cdot 2^{n+2}+3n+8)}$$

$$= \frac{8\cdot 2^{2n+2}+24\cdot 2^{n+2}-72n\cdot 2^{n}-3n^{2}-39n-128}{2^{2n+2}-3\cdot 2^{n+2}+3n+8}$$

$$= 8 + \frac{48\cdot 2^{n+2}-72n\cdot 2^{n}-3n^{2}-63n-192}{2^{2n+2}-3\cdot 2^{n+2}+3n+8}$$

$$= 8 - \frac{(72n-192)2^{n}+3(n^{2}+21n+64)}{2^{2n+2}-3\cdot 2^{n+2}+3n+8}$$

$$< 8$$

Q.E.D.

It is useful to obtain the number of nodes in the quadtree. Letting B and W correspond to the number of BLACK and WHITE, respectively, leaf nodes in the quadtree we have

Lemma 3: The maximum number of nodes in a quadtree having B and W leaf nodes is bounded by $\frac{4}{3}\cdot(B+W)$.

Proof: See Lemma 1 in [10].

We can now prove our main result.

Theorem 2: The average execution time of the Chessboard distance transform computation algorithm is of order B+W.

Proof: From Lemmas 1 and 2 we have that for each side and corner of a BLACK node, GTEQUAL_ADJ_NEIGHBOR, DIST_ADJACENT, GTEQUAL_ CORNER_NEIGHBOR, and DIST_CORNER result in an average maximum of 6+8=14 nodes being visited. There are four sides and corners for each BLACK node. Thus these four procedures contribute $4\cdot B\cdot 14$. From Lemma 3 we have that the number of nodes in the quadtree is bounded by $\frac{4}{3}\cdot(B+W)$. This quantity correlates with the work performed by procedure CHESSBOARD_DIST since each node in the quadtree is visited by the traversal. Summing up these values we have $4 \cdot B \cdot 14 + \frac{4}{3} \cdot (B+W) = \frac{4}{3}(43 \cdot B+W)$.

Q.E.D.

Notice that the amount of work is essentially proportional to the complexity of the image--i.e., to the number of BLACK nodes.

6. Concluding remarks

An algorithm has been presented for computing the Chessboard distance transform for a binary image represented by a quadtree. The algorithm's running time was shown to have an average execution time of order (B+W) where B and W correspond to the number of blocks comprising the objects and the background of the image respectively. It should be noted that the number of BLACK nodes (i.e., the image complexity) dominates the execution time of the algorithm.

The algorithm and the analysis are quite similar to those employed in the computation of the total perimeter [10] and also for the first phase of a connected component labeling algorithm [11] for an image represented by a quadtree. The difference is in part due to the GTEQUAL_CORNER_NEIGHBOR analysis. The similarity should not be surprising because in the former we are searching for adjacencies involving BLACK and WHITE nodes while in the latter we are searching for adjacencies involving BLACK and BLACK nodes. In the case of the distance transform one is also searching for BLACK and WHITE adjacencies except that unlike the perimeter, we do not cease the search when an adjacent BLACK node is found.

Our algorithm is essentially a bottom-up tree traversal. An alternative algorithm might use a top down method; see [13], where a different notion of distance is used. The difficulty with such a method is that it requires the use of more storage so that the distance of a GRAY node to its various sides can be stored. An intermediate approach is one that visits the adjacencies of a BLACK node in a breadth-first fashion rather than depth-first as done here. In such a case we would explore the neighbors of a BLACK node in a ring-like manner. For example, given node P of size 2^S, we would first visit all of the sons of its neighbors that are adjacent to the border of P. If all are BLACK and of the same size, then we start visiting a ring at a further distance. For the image given in Figure 3a this would mean that in order to obtain the distance transform for node 1, we would first visit nodes 2 through 13. If they are all BLACK, we would then visit nodes 14 through 59, etc. This is in contrast with our current approach that locates the closest WHITE node in the northern direction, NE,.... The disadvantage of the breadth-first approach is in the amount of extra bookkeeping that is necessary.

The algorithm can be distinguished from previous work on quadtrees [2,9] in the development of a GTEQUAL_CORNER_NEIGBHOR procedure for locating neighbors along the corner as well as its analysis. This obviates the need for possibly having to invoke GTEQUAL_ADJ_NEIGHBOR more than once (e.g., in Figure 1b, to find the SE neighbor of node C, we locate C's southern neighbor G followed by locating G's eastern neighbor H; however, to find the SE neighbor of node M, we only need to locate M's southern neighbor, Q, since it is larger than M and is also its SE neighbor). The analysis shows that GTEQUAL_CORNER_NEIGHBOR requires somewhat more work than GTEQUAL_ADJ_NEIGHBOR.

Note that procedure CHESSBOARD_DIST attempts to locate all eight neighbors of each BLACK node. Instead, we could check if there is any overlap and avoid invoking GTEQUAL_ADJ_NEIGHBOR or GTEQUAL_CORNER_NEIGHBOR for the overlapping neighbor (e.g., in Figure 1b the eastern neighbor of node I is L which overlaps with the SE neighbor of I). However, the effect of such an improvement is limited since the number of neighbors ranges between five and eight.

It should be clear that the distance transform can be computed for other metrics than the Chessboard. Recall that we chose the Chessboard metric due to its simplicity and the fact that its minimum distance region is a square. The results of Theorem 1 intimate that distance in terms of image width may not be an appropriate measure. Perhaps a node distance is more appropriate. Such a distance measure would reflect the number of nodes that need to be traversed when attempting to make a transition from one BLACK node to its "nearest" WHITE node. This is a subject for future research.

References

1. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley Interscience, New York, 1973.

2. C. R. Dyer, A. Rosenfeld, and H. Samet, Region representation: boundary codes from quadtrees, Computer Science TR-732, University of Maryland, College Park, Maryland, February 1979, to appear in Communications of the ACM.

3. G. M. Hunter, Efficient computation and data structures for graphics, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Princeton, University, Princeton, NJ, 1978.

4. G. M. Hunter and K. Steiglitz, Operations on images using quadtrees, IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 1979, 145-153.

5. G. M. Hunter and K. Steiglitz, Linear transformation of pictures represented by quadtrees, Computer Graphics and Image Processing 10, 1979, 289-296.

6. A. Klinger and C. R. Dyer, Experiments in picture representation using regular decomposition, Computer Graphics Image Processing 5, 1976, 68-105.

7. A. Klinger and M. L. Rhodes, Organization and access of image data by areas, IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 1979, 50-60.

8. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, New York, 1976.

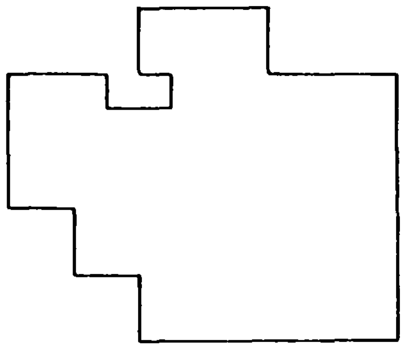
9. H. Samet, Region representation: quadtrees from boundary codes, Computer Science TR-741, University of Maryland, College Park, Maryland, March 1979, to appear in Communications of the ACM.

10. H. Samet, Computing perimeter of images represented by quadtrees, Computer Science TR-755, University of Maryland, College Park, Maryland, April 1979.

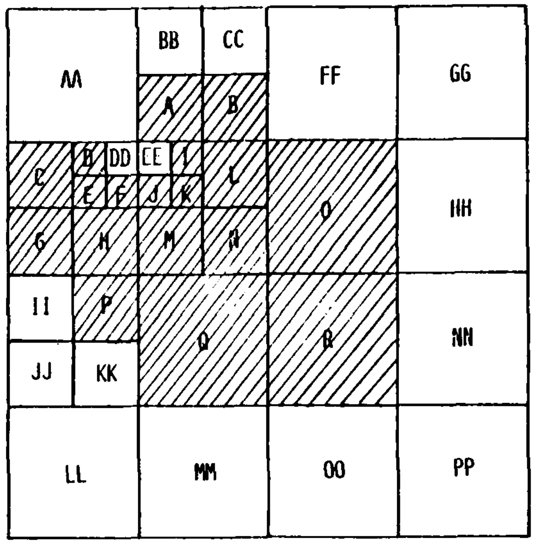
11. H. Samet, Connected component labeling using quadtrees, Computer Science TR-756, University of Maryland, College Park, Maryland, April 1979.

12. H. Samet, A distance transform for images represented by quadtrees, Computer Science TR-780, University of Maryland, College Park, Maryland, July 1979.

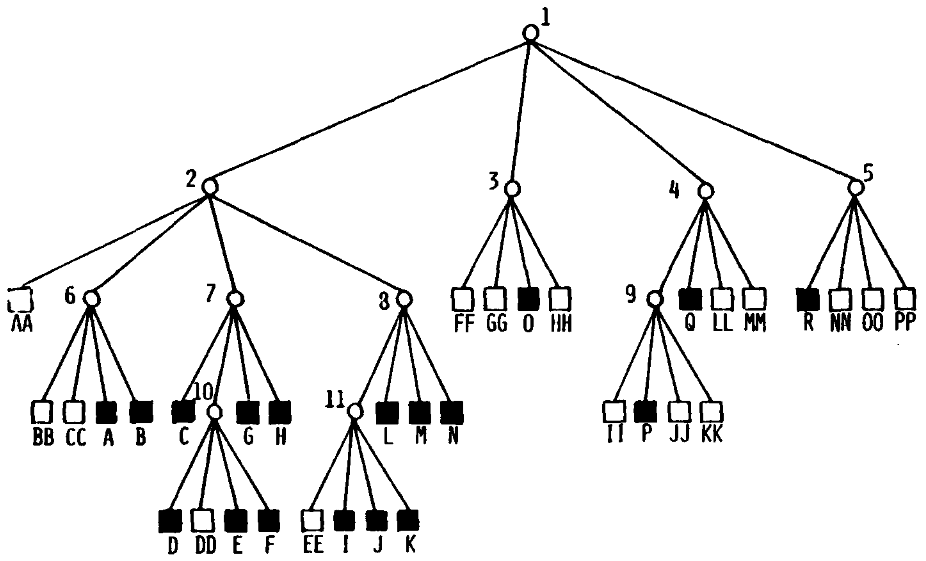
13. M. Shneier, A path-length distance transform for quadtrees, Computer Science TR-794, University of Maryland, College Park, Maryland, July 1979.



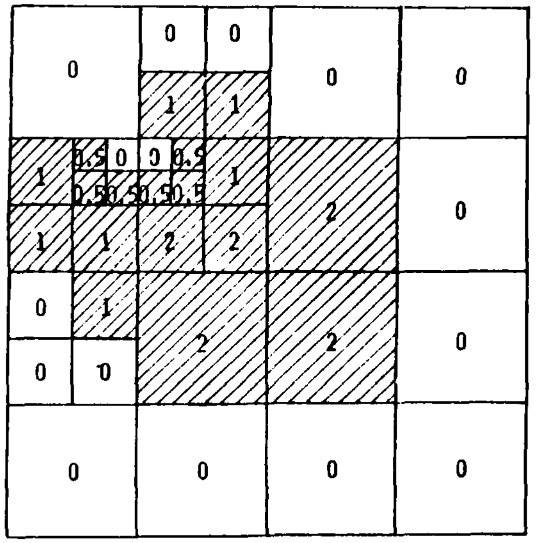
a. Sample image.



b. Block decomposition of the image in (a).



c. Quadtree representation of the blocks in (b).



d. Chessboard distance transform of (b).

Figure 1. An image, its maximal blocks, the corresponding quadtree, and the Chessboard distance transform. Blocks in the image are shaded.

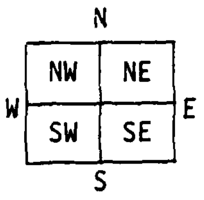
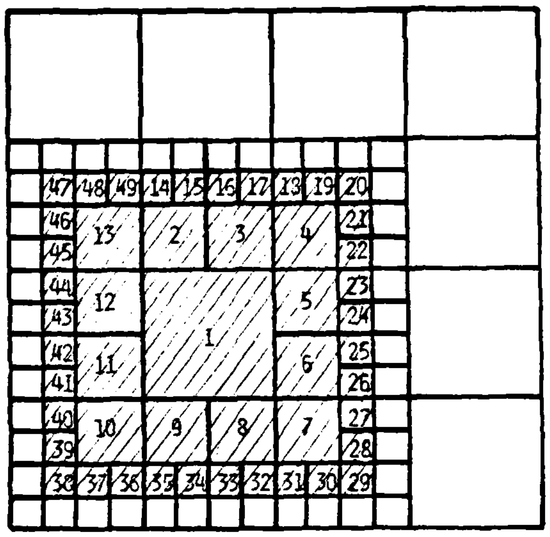
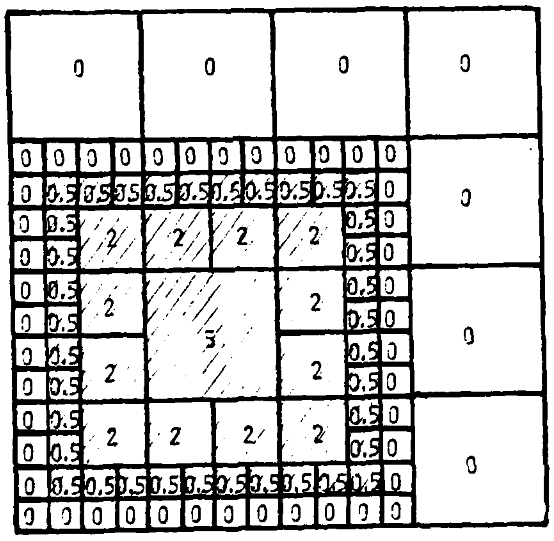


Figure 2. Relationship between a block's four quadrants and its boundaries.



a. Image.



b. Chessboard distance transform of the image in (a).

Figure 3. An image illustrating the maximum number of nodes that need to be visited when computing the Chessboard distance transform value for node 1.

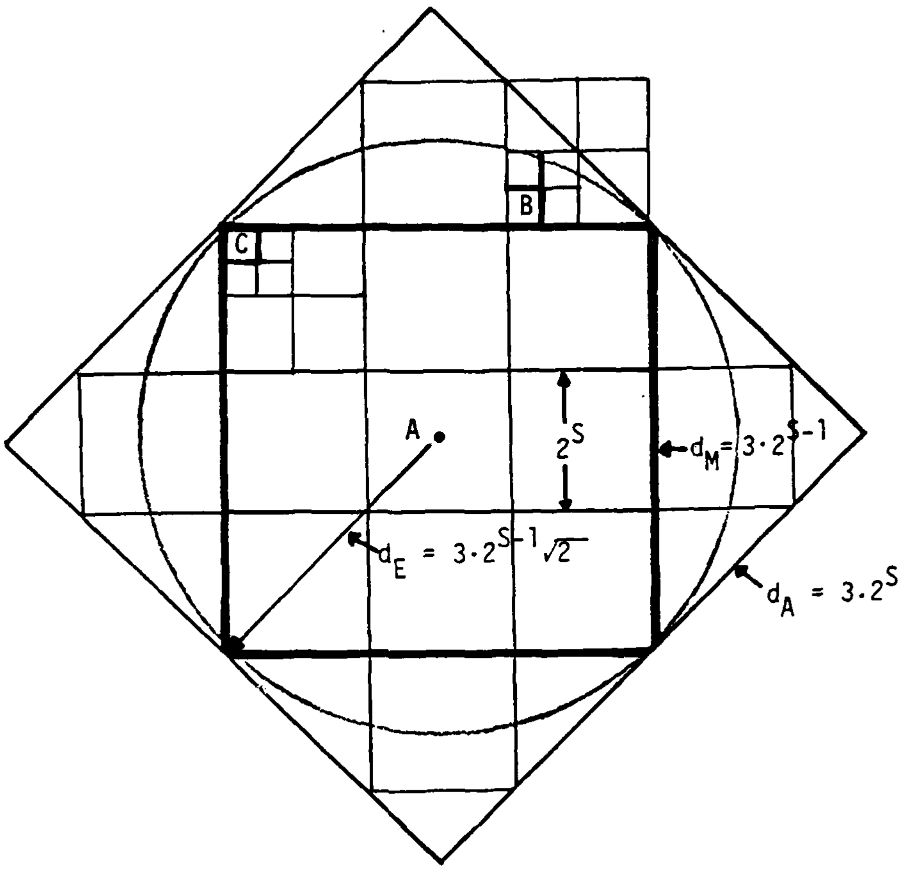


Figure 4. Regions within which the closest WHITE node to BLACK node A must lie for several metrics.

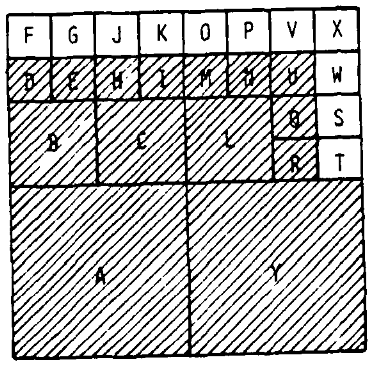


Figure 5. Image illustrating the worst case for N and NE neighbors when n=3.

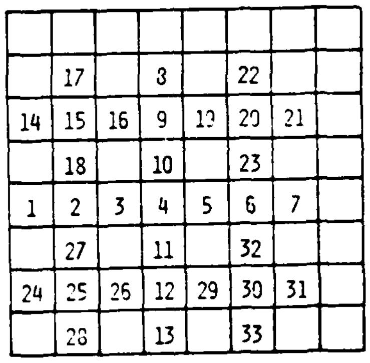


Figure 6. Sample grid illustrating nodes at level Ø whose nearest common ancestor is at level ≥ 2 when attempting to locate a NE neighbor.

UNCLASSSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A084 290	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A DISTANCE TRANSFORM FOR IMAGES REPRESENTED BY QUADTREES		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER TR-780
7. AUTHOR(s) Hanan Samet		8. CONTRACT OR GRANT NUMBER(s) DAAG-53-76C-0138
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Center University of Maryland College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Night Vision Laboratory Fort Belvoir, VA 20060		12. REPORT DATE July 1979
		13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image processing
Pattern recognition
Quadtrees
Distance transforms

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The concept of distance used in binary array representations of images is adapted to a quadtree representation. The Chessboard distance metric is shown to be particularly suitable for the quadtree. A Chessboard distance transform for a quadtree is defined as the minimum distance in the plane from each BLACK node to the border of a WHITE node. An algorithm is presented which computes this transform by only examining the BLACK node's adjacent and abutting neighbors and their progeny. Analysis of the algorithm shows

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

that its average execution time is proportional to the number of leaf nodes in the quadtree.